1974-03

A General Purpose Microcontroller

Brubaker, Raymond H. Jr.

http://hdl.handle.net/10945/68410

Appendix 3

A GENERAL PURPOSE MICROCONTROLLER

Raymond H. Brubaker, Jr.
Assistant Professor of Computer Science
Naval Postgraduate School
Monterey, California

March, 1974

Background

This paper describes a simple programmable control unit or "microcontroller." It was designed to provide the necessary sequence of control signals for many digital applications including (1) the interface between a "floppy disk" and a small computer, (2) a control unit for the IBM System/360 multiplexor channel, and (3) the serial telecommunications interface for the NPS Ring Network (the Ring Interface).

The applications of programmable controllers are almost limitless. They have become a cost-effective solution to digital control with the advent of low-cost semiconductor read-only-memory (ROM). The advantages of the microprogrammed approach are summarized below:

1. Structured designs. A more structured overall design can be achieved as random logic is reduced.
2. Adaptability. A given design can be easily changed to meet varying external needs. (Build one disk controller, and change the program to suit different computers.)
3. Debugging and update. Changes are made by altering the control store rather than rewiring.
4. Faster implementation. Designs go from conception to prototype faster with a standard, programmable control unit.
5. Fault diagnosis. Diagnostic aids can be programmed into the controller itself.

A General Purpose Microcontroller

The microcontroller described here functions basically like a small computer with a 1.1 microsecond instruction cycle and at least 256 words of reprogrammable ROM for program storage. Only four different operation codes are used: Jump Unconditionally (JU), Jump on True input (JT), Jump on False input (JF), and Output (OUT). JU causes an unconditional jump to any location on any of the 256 word pages of memory. JT tests one of 32 inputs to the controller and jumps to the specified location on the current page if the test is <u>true</u>; otherwise the next sequential instruction is executed. JF is similar with the jump occuring when the selected input is <u>false</u>. OUT briefly (100 nanoseconds, nominal) strobes one of 32 control lines and displays an 8-bit data word concurrently.

[illegible] p

[illegible] page used
this application

In summary, this four-instruction computer can generate a sequence of control signals, with or without data, to operate 32 distinct "devices." This sequence can be altered or repeated using jumping instructions which may be conditioned by the state of up to 32 input variables. The detailed instruction formats are shown in figure 1.

	15 13	12 8	7 0
OUT:	0 0 0	unit select	data out
JU:	1 0 0	page number	location
JT:	1 0 1	input select	location
JF:	1 1 0	input select	location

(Note that bits 0 through 12 are stored in complement form.)

Figure 1. Instruction Format

Machine Architecture

The architecture of the controller is shown in block form in figure 2. For purposes of discussion, it can be divided into four basic units: memory, instruction counter, input selector (multiplexer), and output selector (decoder). A schematic is attached to this paper.

Memory.

Instruction memory is provided by pages of 16-bit words with 256 words per page. Up to 32 pages may be attached although it appears that many complex applications can be handled with one or two-page controllers. The upper three bits of a ROM word feed a decoder to yield eight distinct opcode lines (only four are currently used). The next five bits assume a different selection role depending on the operation (see figure 1). The lower eight bits provide the address in jumping instructions and the parallel data for output operations.

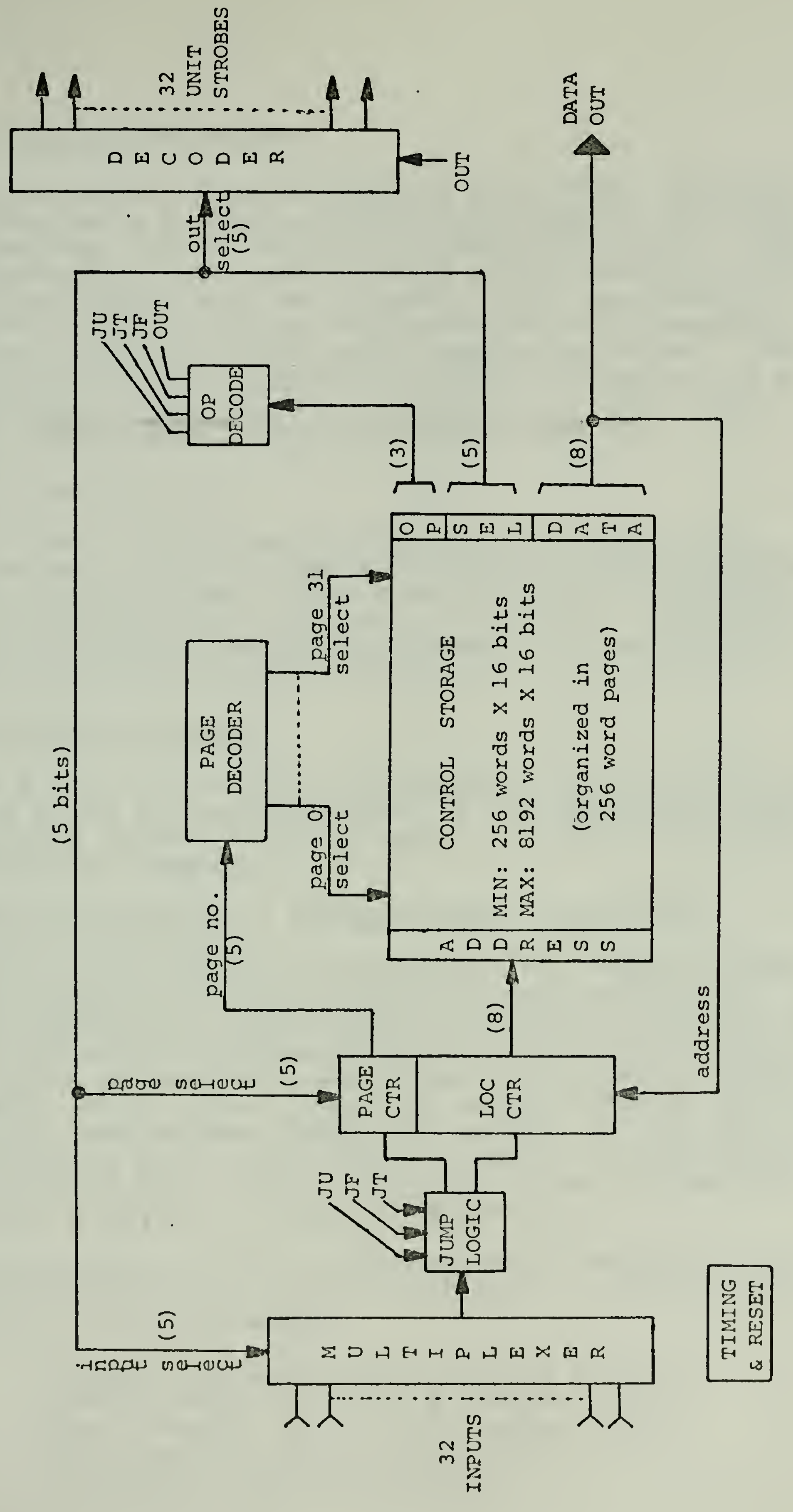


Figure 2. Microcontroller Architecture

The Instruction Counter.

The instruction counter consists of an 8-bit location counter and a 5-bit (maximum) page counter. After OUT operations, the IC is simply incremented by one. During JU operations, the lower 13 bits of the instruction are parallel-loaded into the counter. On a successful JT or JF operation, 8 bits are loaded into the location counter while the page number remains unchanged. Note that it is not possible to jump out of a page using a conditional jump, but it is possible to "fall" off a page during normal incrementation of the IC.

The Input Selector.

Bits 8 to 12 of JT and JF instructions are used to select one of the 32 inputs for testing. This selected value, coming from a 32-to-1 multiplexor, is fed into the branching logic along with the op-code. Together they are used to control the loading and incrementing of the page and location counters.

The Output Selector.

Bits 8 to 12 of an OUT instruction select one of 32 output lines using a 5-to-32 decoder. A 100 nanosecond pulse is placed on that line just prior to selecting the next instruction from ROM.

Sample Application: A Traffic Signal Controller

Consider the problem of controlling the traffic signals in a typical 4-way intersection. Let's assume that North-South (NS) is the favored direction, that is, the NS light will stay green unless the East-West (EW) walk button is pressed or a car drives over a sensor buried beneath the EW lanes. Just for variety (and to make the control problem more difficult) we will set the NS lights to flashing yellow, and the EW lights to flashing red during late night hours. Figure 3 presents a possible control sequence for such an intersection.

Implementation with the microcontroller requires that we first define the input/output characteristics of the devices to be controlled:

> Traffic lights: these will have a 2-bit binary color input and a "change" input. When "change" is 1, a new "color" value is accepted and displayed. (00-off, 01-green, 10-yellow, 11-red)

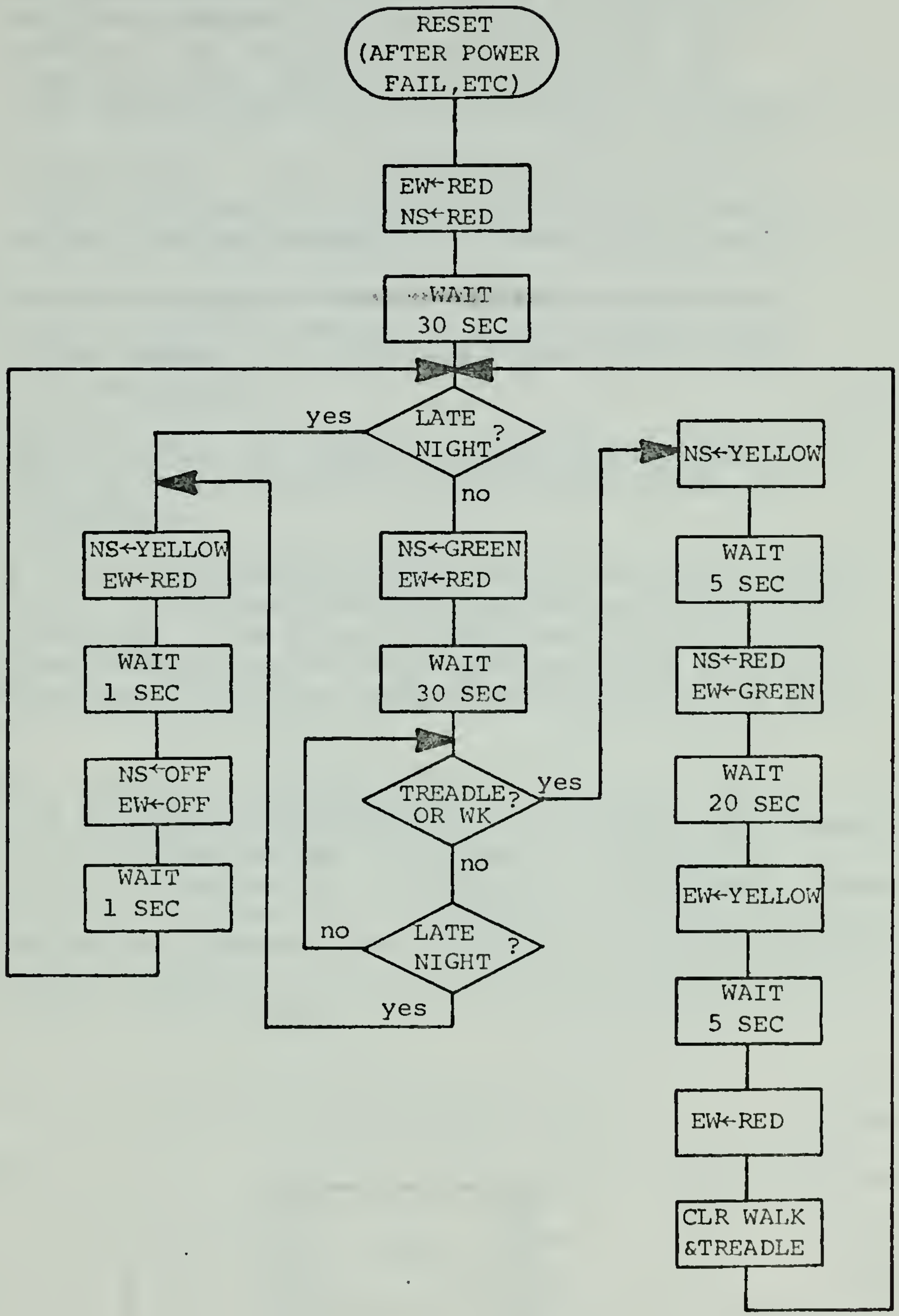
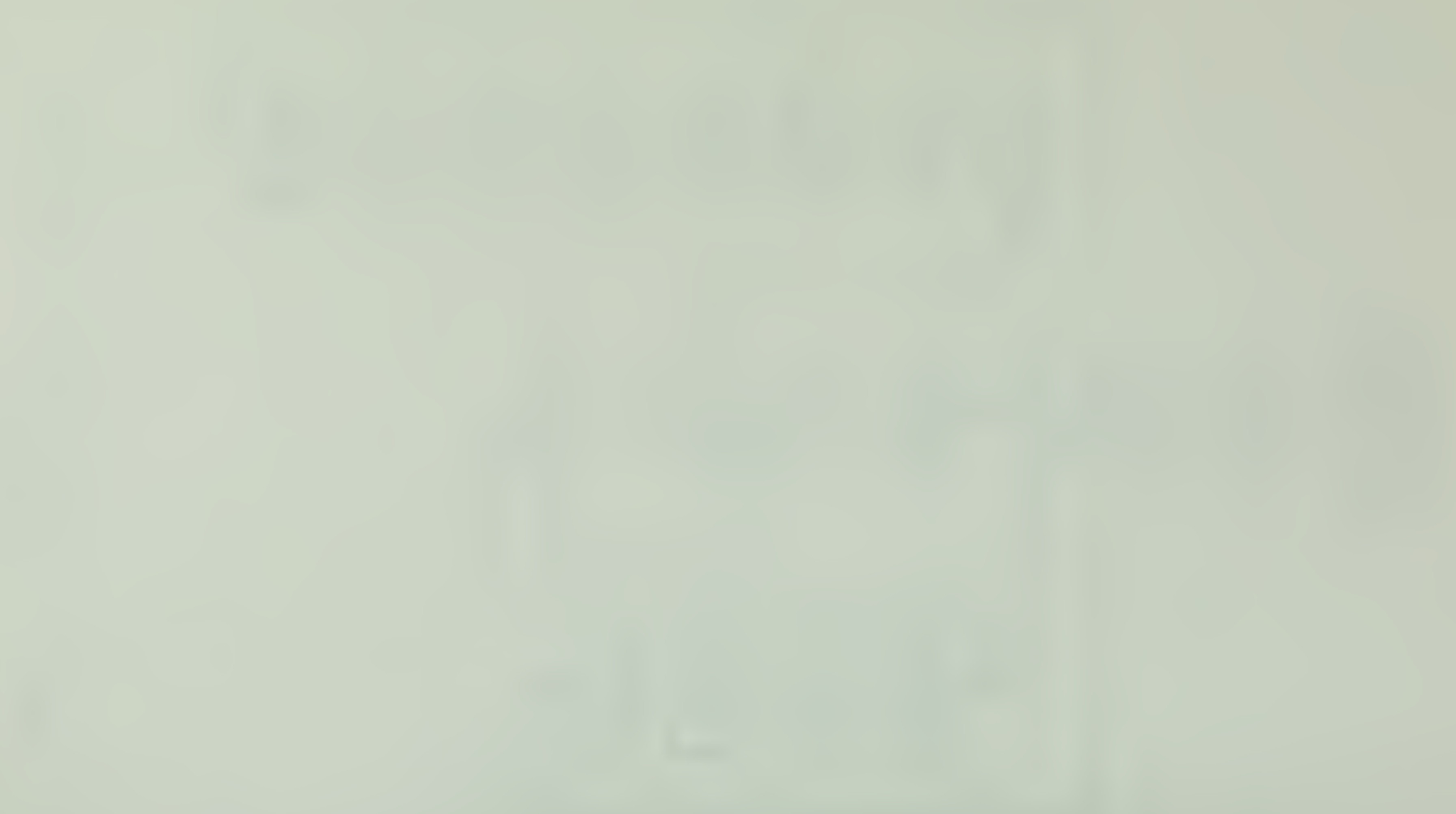


Figure 3. Control Sequence for Traffic Controller



Timer: an 8-bit binary counter which can be loaded to any value up to 255 and then self-decrements to zero at one count per second. An "expired" latch records the fact that the counter has reached zero. The latch resets when the counter is loaded.

Late night: a real-time clock which keeps track of time-of-day and provides a true output during preselected late-night hours.

Treadle: Any of a group of sensors beneath the EW traffic lanes. A 1 is latched when a car passes over and held until reset by the controller.

Walk button: As above, but indicates the request of a pedestrian to cross in the EW direction.

Figure 4 shows the hookup of the devices to the microcontroller. Note that four inputs are required for decision-making, five outputs (strobes) to reset and control the various units, and the data bus is used for setting the timer and selecting the color of both signals.

A symbolic program to implement the traffic signal controller with the microcontroller is given in figure 5. Symbols are defined using the "equal" symbol (=). Comments are indicated by a slash (/). Statement labels are set off by a colon (:). Fields of the instructions are separated by commas. The asterisk (*) is used to indicate the location of the current instruction for single-instruction loops.

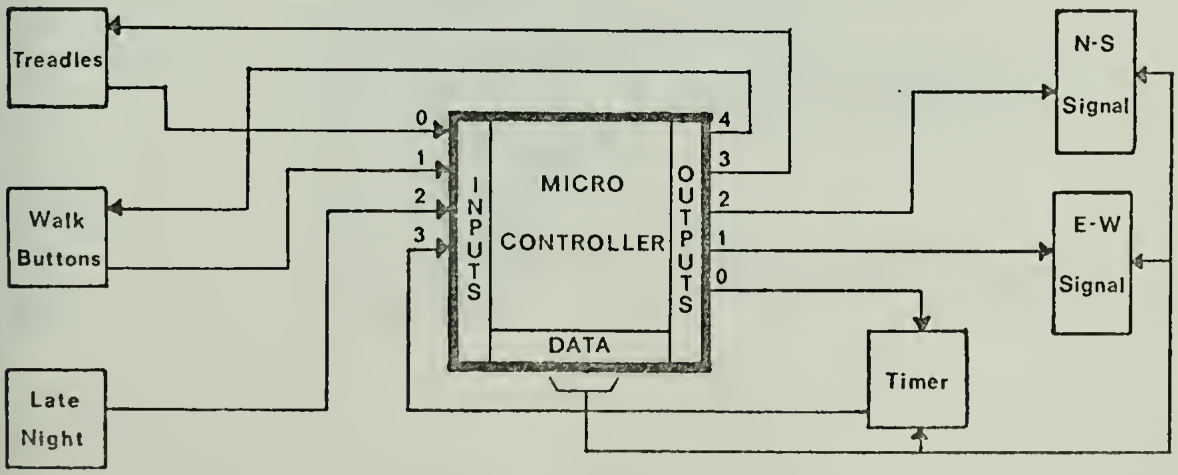


Figure 4. Traffic Controller Schematic

```
ADDR                        STATEMENT

          / INPUT DEFINITIONS
                  TREADLE=0; WALK=1; NIGHT=2; TIMEOUT=3

          / OUTPUT DEFINITIONS
                  SETTIMER=0; EW=1; NS=2; CLRTREAD=3; CLRWK=4
                  OFF=0; GREEN=1; YELLOW=2; RED=3

          / COME HERE AFTER POWER FAIL OR OTHER RESET
 0        RESET:  OUT,NS,RED; OUT,EW,RED /BOTH LIGHTS RED
 2                OUT,SETTIMER,30
 3                JF,TIMEOUT,*   / WAIT 30 SECONDS

          /MAIN LOOP: CHECK FOR LATE NIGHT
 4        MAIN:   JT,NIGHT,FLASH
 5                OUT,NS,GREEN; OUT,EW,RED
 7                OUT,SETTIMER,30
 8                JF,TIMEOUT,*    / WAIT 30 SECONDS

 9        CHECK:  JT,TREADLE,CHANGE
10                JT,WALK,CHANGE      /  LOOP UNTIL EVENT
11                JF,NIGHT,CHECK      /

          / LATE AT NIGHT....FLASH THE LIGHTS
12        FLASH:  OUT,NS,YELLOW; OUT,EW,RED
14                OUT,SETTIMER,1
15                JF,TIMEOUT,*   / WAIT ONE SECOND
16                OUT,NS,OFF; OUT,EW,OFF
18                OUT,SETTIMER,1
19                JF,TIMEOUT,*   / WAIT ONE SECOND
20                JU,MAIN  / GO SEE IF IT'S MORNING YET

          / CHANGE EW TO GREEN IN RESPONSE TO WALK BUTTON
          / OR TREADLE
21        CHANGE: OUT,NS,YELLOW    / SEQUENCE FROM NS GREEN
22                OUT,SETTIMER,5   / TO EW GREEN
23                JF,TIMEOUT,*
24                OUT,NS,RED; OUT,EW,GREEN
26                OUT,SETTIMER,20
27                JF,TIMEOUT,*   / WAIT 20 SECONDS
28                OUT,EW,YELLOW  / SEQUENCE EW TO RED
29                OUT,SETTIMER,5
30                JF,TIMEOUT,*
31                OUT,EW,RED
32                OUT,CLRTREAD,0   / RESET TREADLES
33                OUT,CLRWK,0      / RESET WALK BUTTON
34                JU,MAIN
```

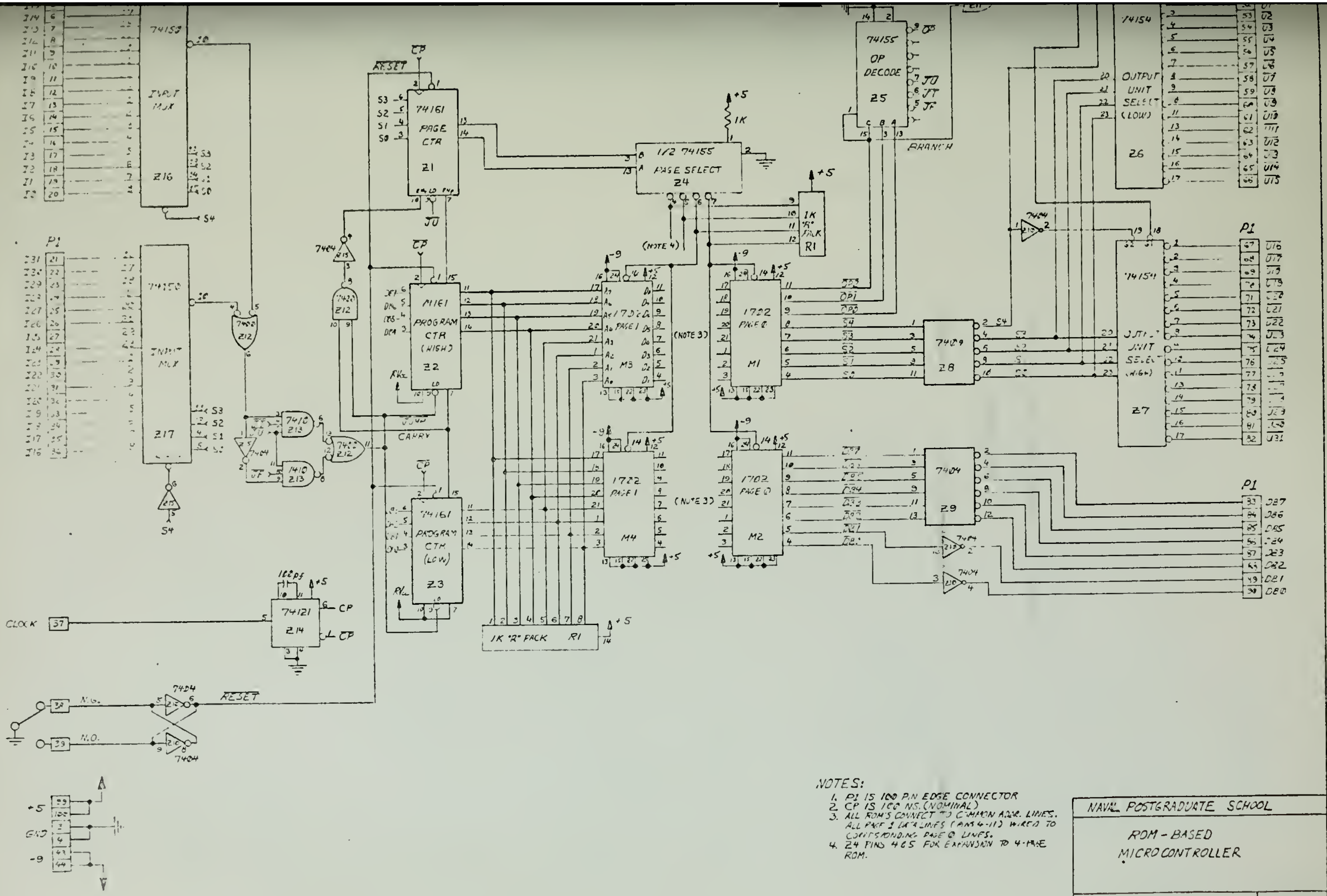
Figure 5. Symbolic Traffic Controller Program

The program requires 35 words of ROM storage. The ROMs are normally programmed using binary data recorded on paper tape. These tapes could be generated either manually, converting each instruction in figure 5 to binary, or with the help of a symbolic assembler. A symbolic assembler which runs on Intel's 8008 microcomputers is available. The statements accepted by this assembler are more concise (less readable at first) to make the assembler faster and smaller, and to make the speed of a teletype more bearable. The assembler makes two passes over the source program and produces a paper tape suitable for programming 1702A ROMs.

Possible Extensions

Just as in higher-level and assembler-level programming, the microprogrammer finds it necessary to repeat the same sequence of steps at several points in a control sequence. To save space in the control memory (ROM) a subroutine capability could be added. This would require the additional circuitry to stack the current IC value (CALL) and later restore a stacked IC value (RETURN). Note that this will degrade performance due to the two extra instruction cycles required to invoke the shared routine.

With 1702A ROMs the instruction cycle is limited to about 1.1 microseconds. Using newer fusible-link ROMs (or even masked ROMs in production applications) combined with high speed logic a cycle less than 200 nanoseconds is easily attainable.



NOTES:

1. P1 IS 100 PIN EDGE CONNECTOR
2. CP IS 100 NS.(NOMINAL)
3. ALL ROM'S CONNECT TO COMMON ADDR. LINES. ALL PAGE 1 DATA LINES (PINS 4-11) WIRED TO CORRESPONDING PAGE 0 LINES.
4. Z4 PINS 4&5 FOR EXPANSION TO 4-PAGE ROM.

NAVAL POSTGRADUATE SCHOOL

ROM-BASED
MICROCONTROLLER

R.H. Bredaker jr.

6 MARCH 74

APPENDIX 4

MOTOROLA BIPOLAR LSI UNIVERSAL POLYNOMIAL GENERATOR

MOTOROLA Semiconductors

MC8503P

BIPOLAR LSI

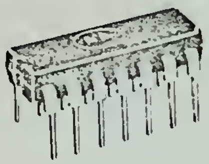
UNIVERSAL POLYNOMIAL GENERATOR

UNIVERSAL POLYNOMIAL GENERATOR (UPG)

The MC8503 Universal Polynomial Generator (UPG) is used in serial digital data handing systems for error detection and correction. The serial data stream is divided by a selected polynomial and the division remainder is transmitted at the end of the data stream as a Cyclic Redundancy Check Character (CRCC). When the data is received the same calculation is performed. If there were no errors in transmission, the new remainder will be zero.

The MC8503 offers four of the more common polynominals for error detection techniques including a read forward and reverse on the CRCC-16 and CRCC-CCITT polynomial functions. These polynomials can be generated by changing the binary select codes as shown in Figure 1.

- Four Unique Polynomial Codes in One Package
- Compatible with TTL
  Maximum Fan-Out = 1 TTL Load
- Data Rate = 5 MHz Typical
- Total Power Dissipation = 400 mW Typical
- +5.0-Volt Operation

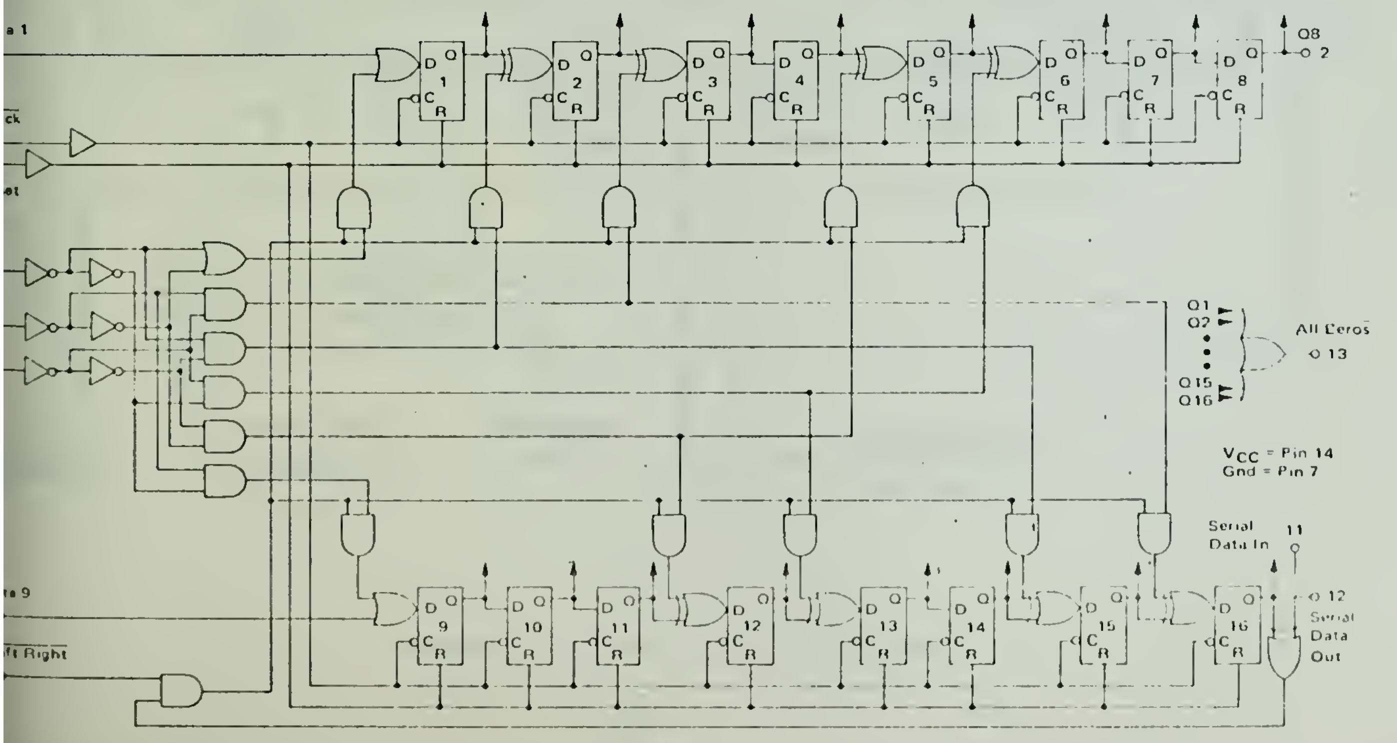


PLASTIC PACKAGE
CASE 646

FIGURE 1 – AVAILABLE POLYNOMIALS

CODE SELECT X	CODE SELECT Y	CODE SELECT Z	POLYNOMIAL	
0	0	0	CRCC 16 (Fwd)	$x^{16} + x^{15} + x^2 + 1$
0	0	1	CRCC 16 (Bkwd)	$x^{16} + x^{14} + x + 1$
1	1	0	CRCC CCITT (Fwd)	$x^{16} + x^{12} + x^5 + 1$
1	1	1	CRCC-CCITT (Bkwd)	$x^{16} + x^{11} + x^4 + 1$
0	1	0	LRCC-16	$x^{16} + 1$
1	0	1	LRCC 8	$x^8 + 1$

LOGIC DIAGRAM



VCC = Pin 14
Gnd = Pin 7